Neural Comput & Applic (2004) 13: 90–98
DOI 10.1007/s00521-003-0390-z

ORIGINAL ARTICLE

Hanh H. Nguyen · Christine W. Chan

Multiple neural networks for a long term time series forecast

Received: 17 February 2003 / Accepted: 27 October 2003 / Published online: 18 December 2003

Abstract The artificial neural network (ANN) methodology has been used in various time series prediction applications. However, the accuracy of a neural network model may be seriously compromised when it is used recursively for making long-term multi-step predictions. This study presents a method using multiple ANNs to make a long term time series prediction. A multiple neural network (MNN) model is a group of neural networks that work together to solve a problem. In the proposed MNN approach, each component neural network makes forecasts at a different length of time ahead. The MNN method was applied to the problem of forecasting an hourly customer demand for gas at a compression station in Saskatchewan, Canada. The results showed that a MNN model performed better than a single ANN model for long term prediction.

Keywords Multiple neural networks · Time series forecasting

1 Introduction

A common problem with the time series forecasting model is the low accuracy of long term forecasts. The estimated value of a variable may be reasonably reliable for the short term future, but for the longer term future, the estimate is likely to become less accurate. There are several possible reasons to account for this increasing inaccuracy. One reason is that the environment in which the model was developed has changed over time. Therefore, the input valid at a given time interval does not in fact have an influence on the output relevant for a time interval quite some distance away in the future. Another reason is that the model itself was not well developed. The inaccuracy arises due to immature training or a lack of appropriate data for training. The trained model may cover the surrounding neighbourhood of data but fails to model cyclic changes of trend or seasonal patterns of data. The third cause of inaccuracy is propagation errors that grow during recursive model predictions. To predict p step ahead, a one-step-ahead neural network is used recursively p times. Every model is likely to be associated with an error. For long-term predictions, this error is accumulated and can increase beyond an acceptable threshold.

To address the third problem, we propose a multiple neural network (MNN) model that combines short-term and long-term neural networks to accommodate a wide range of prediction terms. The MNN approach deals with the third problem by reducing the number of recursions necessary. In this approach, several neural networks built to predict from short- to long-term are combined into one model. Hence, the objective of this research is to investigate whether grouping neural networks into a model improves the forecast performance of a single neural network model in long-term forecasting.

This paper is organised as follows. Sect. 2 provides some background literature on neural networks in time series forecasting and the MNN approaches. Sects. 3 and 4 describe the design and implementation of the proposed MNN approach. Sect. 5 presents a case study using the proposed model to predict the hourly flow rate at a gas station in Saskatchewan, Canada. Sect. 6 describes the conclusions and future work.

2 Background literature

2.1 Artificial neural networks (ANNs) in time series forecasting

Researchers have made several comparisons between different types of neural networks and ARIMA models. For example, a multi-layer perceptron (MLP) has been

H. H. Nguyen · C. W. Chan (✉)
Faculty of Engineering, University of Regina,
Regina SK S4S 0A2, Canada
E-mail: Christine.Chan@uregina.ca

shown to be equal to a nonlinear auto-regression model or a Jordan net is equal to a non-linear moving average model [4]. Since a MLP, and in particular, a back-propagation neural network has been adopted in this study, the MLP is discussed as follows.

The output vector of a MLP can be expressed by the following equation:

$$\vec{y} = \left(\sum_{j=1}^{k} w_{jl}^{o} \sigma \left(\sum_{i=1}^{n} w_{ij}^{h} x_i + b_j^h \right) + b_l^o \right), \; l = 1 \ldots m,$$

where $\vec{x}$ is the input vector, $\vec{y}$ is the output vector, $\vec{w}^h$, and $\vec{w}^o$, and $\vec{b}^h$ and $\vec{b}^o$are the weights and biases of hidden and output layers, n and m are the sizes of input and output vectors, k is the number of hidden units, and σ is a non-linear function. If the input vector contains p previous sequence elements and the output vector contains the current element, the equation can be replaced with

$$x(t) = f(x(t-1), x(t-2), ..., x(t-p)) + \varepsilon(t),$$

where $\epsilon(t)$ is the noise term, x is the time series and t is the time index for the current time. This equation is similar to the auto-regression equation in an auto-regression model, with the exception that function f is non-linear instead of linear. Back-propagation neural networks and other feed-forward neural networks do not include the moving-average elements.

Some limitations of ARIMA models include their linearity and a requirement of stationarity. Neural networks are not constrained by the assumption of stationarity and can model the more complex characteristics of a time series. However, due to a high degree of freedom, neural networks usually require a large set of training data. Moreover, over-fitting of the data and non-uniqueness are common problems with neural networks.

Applications of neural networks and time series modelling for prediction in various economic, scientific and industrial problems have produced good research results. Some examples include Walczak [15] who achieved up to a 70% forecast accuracy on currency exchange rates, as well as Lertpalangsunti and Chan [9] and Lertpalangsunti et al. [10] who succeeded in producing utility demand forecasts.

Tang et al. [14] made a quantitative comparison between ANNs and Box-Jenkins methods for time series forecasting. They concluded that Box-Jenkins models are slightly better in short term forecasting while neural networks are better for long term forecasting. Moreover, for a short memory time series, neural networks appear to be superior to Box-Jenkins models. The two phrases "short term forecasting" and "short memory time series" can be clarified as follows. The "term" refers to the period of time in the future for which a prediction is made, and "memory" refers to the correlation of a prediction back to x previous intervals of time. Hence, for short memory predictions, x is small and for long memory predictions, x is large.

In Tang et al. [14], Box-Jenkins models were identified using TIMESLAB's model indentification macro. Three experiments were set up to demonstrate three sets of time series data with different characteristics. The airline passenger data set was used in Tang et al. [14] and Faraway and Chatfield [6]. The data set has a long memory, an apparently increasing trend and a seasonal pattern. The other two sets of data are on domestic car sales and foreign car sales. They show some roughly seasonal patterns. The former represents a short memory time series with an irregular trend while the latter shows a relatively smooth increasing trend. For the set of data on domestic car sales with an irregular trend, neural networks worked better than the Box-Jenkins model. By comparing the forecast results for the three sets of time series data, Tang et al. concluded that as the complexity of the time series increases, the neural network model gives a better performance than the Box-Jenkins model.

Tang et al. calculated the total sum square error (TSS) of the 24 hour period forecast as a measurement of performance. The TSS for the two techniques were compared along three dimensions: (1) the amount of data used (2) the number of forecasted periods into the future (3) the number of input variables. The following conclusions were drawn:

- The amount of data can affect the performance of the forecast technique, and more training data typically means a more accurate forecast. However, even with a small amount of time series data as an input, the neural network can perform reasonably well while the Box-Jenkins model cannot. This can be regarded as one of the advantages of neural networks over Box-Jenkins models.
- The results of the experiments on the three sets of data in [14] show that the Box-Jenkins model outperforms the neural network for the selected structures and training methods for forecasts of one period and six periods ahead. On the other hand, for forecasts of 12 periods and 24 periods ahead, the neural network is superior. The relative performance of the neural network improves as the forecast horizon increases, which suggests that an ANN is a better choice for long-term forecasting.
- When the number of input variables increases, the forecasting ability of the neural network improves. It is likely that more accurate forecasts are produced when more information has been provided by the increased number of input variables. However, there is a trade-off between the accuracy of the model and the model complexity in terms of the number of input variables. To enhance accuracy of the forecast, the size of the training set should be relatively large when the number of input variables increases.

More recent comparisons between neural network and Box Jenkins methods can also be found in [3, 6].

2.2 Multiple neural network approaches

Several researchers have attempted to use multiple neural networks to improve model accuracy. This section will review some of the approaches that are not necessarily related to time series modelling, but which explore different aspects related to multiple neural networks.

For a given prediction problem, several neural network solutions can be obtained. The network resulting in the least testing error is usually chosen. However, the network may not be the optimum when it is applied to the whole population. Hashem et al. [7] proposed using optimal linear combinations of a number of trained neural networks instead of using a single best network. Each component network can have a different architecture and/or training parameters. Optimal linear combinations are constructed by forming weighted sums of the corresponding outputs of the networks. The combination weights are selected to minimise the mean squared error with respect to the distribution of random inputs. Combining the trained networks may help integrate the knowledge required by the component networks and thus improve model accuracy. From a neural network perspective, combining the corresponding outputs of a number of trained networks is similar to creating a large neural network in which the component neural networks are sub-networks operating in parallel, and the combination weights are the connection weights of the output layer.

The ensemble neural network system introduced by Hashem et al. can be used for prediction problems only. Cho and Kim [2] presented a method using fuzzy integrals to combine MNNss for classification problems. Unlike other methods such as the majority voting[1] or the Borda count[2], the proposed method not only combines the results from the component networks but also considers the relative importance of each network. For each new input datum, each trained component neural network calculates the degree of certainty h that the object belongs to a class. Next, the degree of importance g of each component network in recognition of a class is calculated. The fuzzy integral of each class is then computed based on the values of h and g. Finally, the class with the largest integral value is chosen as the output class.

Lee [11] focused more on the data and its distribution. Lee introduced a multiple neural network approach in which each network ha ndles a different set of the input data with different weights. In this approach, a sub-network is created when the main network has little confidence in its decision. The main network and sub-network share the same input vector but each network has its own hidden and output layers. The main network handles most of the cases while sub-networks handle more or less irregular parts of the data.

The output of the system is selected among multiple outputs from the neural networks using a preference vector.

Kadaba et al. [8] developed a MNN system to improve accuracy by decreasing the input and output cardinalities. They used back-propagation self-organising networks to compress data records and then used the concentrated low-cardinality data records to feed a classification neural network.

The work by Duhoux et al. [5] is most relevant to the study area of this paper. Duhoux et al. compared two methods for long-term prediction with neural network chains. The classical method makes predictions one-step-ahead recursively. In this method, only a single one-step-ahead neural network is trained and it is used iteratively p times to predict p step ahead. The input is shifted correspondingly at each iteration step. The proposed method, on the other hand, uses p different trained neural networks with different sizes of input vectors ranging from 1 to p. The input of a network is the same as that of the previous network plus the predicted output from the previous network. However, Duhoux et al. also reported some disadvantages of the proposed method: (1) the model requires a large amount of neural networks and input variables, and (2) a priori knowledge about the signal tendencies is not used. This work formed the basis of our proposed MNN system.

3 MNN system design

The MNN approach proposed in this paper aims to improve upon the classical recursive one-step-ahead neural network approach. The objective of the new approach is to reduce the number of recursions needed while not requiring too many component neural networks. A MNN model is a group of ANNs working together to perform a task. Each ANN is developed to predict a different time period ahead. The prediction terms are powers of 2, that is, the first ANN (0-ordered) predicts 1 unit ahead, the second (1-ordered) predicts 2 units ahead, the third (2-ordered) predicts 4 units ahead, and so forth. Hereafter an ANN that predicts 2^n units ahead is referred to as an n-ordered ANN.

There are two reasons to support the choice of binary exponential. First, big gaps between two consecutive neural networks are not desirable. The smaller the gaps are, the fewer steps the model needs to take in order to make a forecast. Forecasting is a process of reducing the lead time to zero and smaller. For example, to forecast one hour ahead, we reduce the lead time from 1 (as in x_{t+1}) to 0 (as in x_t) and smaller (as in x_{t-1}) as follows: $x_{t+1} = f_1(x_t, x_{t-1})$. Similarly, it takes five steps to reduce a lead of 25 to 0 in the base 3 exponential:

$$25 \xrightarrow{f_2} 16 \xrightarrow{f_2} 7 \xrightarrow{f_1} 4 \xrightarrow{f_1} 1 \xrightarrow{f_0} 0$$

[1]The majority voting rule chooses the classification made by more than half of the networks.

[2]The Borda count of a class is the sum of the number of classes ranked below that class by each network. The class of which the Borda count is the largest is chosen.

while it takes only three steps to reduce a lead time of 25 from order 4 to order 0 in the binary exponential system: $25 \xrightarrow{f_4} 9 \xrightarrow{f_3} 1 \xrightarrow{f_0} 0$. Secondly, the binary exponential does not introduce bias on the roles of networks. A higher exponential model tends to use more lower-ordered neural networks in order to propagate ahead. Hence, the burden of prediction is imposed on the lower-ordered neural networks, which reduces the effectiveness of the MNN system. For example, the following sequences are used to reduce a lead time of 29 to 0 in the base 5 exponential and binary exponential: $29 \xrightarrow{f_2} 4 \xrightarrow{f_0} 3 \xrightarrow{f_0} 2 \xrightarrow{f_0} 1 \xrightarrow{f_0} 0$, and $29 \xrightarrow{f_4} 13 \xrightarrow{f_3} 5 \xrightarrow{f_2} 1 \xrightarrow{f_0} 0$. In this example, the base 5 exponential system uses the second-ordered neural network once and the zero-ordered neural network four times while the binary exponential uses each order (4, 3, 2, 0) only once. It can be noted that the various forecasts at different lead times may not be consistent with each other.

A MNN prediction model can be viewed as a single partially connected ANN as illustrated in Fig. 1. Figure 1 shows a sample MNN with two sub-ANNs to predict 3 units ahead. However, with a higher lead time, the combination will become much more complex with several layers and connections among sub-ANNs. Since the weights of sub-ANNs and the weights connecting sub-ANNs in a combined ANN are dependent on each other, they should be estimated at the same time. This requirement means it takes a long time to train a complex combined ANN. In contrast, a MNN breaks down the training into sub-ANNs and trains them separately.

To make a prediction, the neural network with the highest possible order is used first. For example, to predict 7 units ahead, a 2-ordered neural network is used first. Assume that the time at present is t. The values of x_t and x_{t-1} are already known. We wish to predict x_{t+7}. Let us denote the function that an n-ordered network models as f_n and assume that every network has two input variables, then the value of the output 7 units ahead is computed as follows:

$$\begin{aligned} x_{t+7} &= f_2(x_{t+3}, x_{t+2}) \\ &= f_2(f_1(x_{t+1}, x_t), f_1(x_t, x_{t-1})) \\ &= f_2(f_1(f_0(x_t, x_{t-1}), x_t), f_1(x_t, x_{t-1})) \end{aligned}$$

The training of individual component neural networks can be dependent or independent on the training of the other networks. In the development of a MNN, it may be necessary to implement multi-step validation. One-step-ahead validation does not take into account the model's sensitivity to errors that arise due to multi-step predictions [12]. In our MNN tool, multi-step validation was implemented, but the validation window for each ANN is different. The validation error of the n-ordered network is calculated as the average of root mean square errors (RMSE) of the (2^n)-step-ahead to the $(2^{n+1}-1)$-step-ahead. In order to calculate these steps, a higher ordered ANN needs to use the prediction values of the lower ordered ANNs.

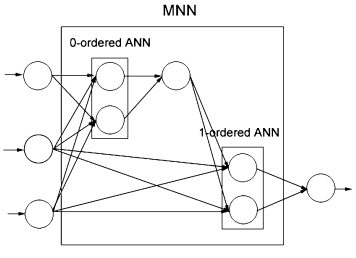

Fig. 1 A sample MNN model

4 MNN system implementation

The MNN system was created to assist in the development of MNN applications. The implementation and usage details of the tool are described as follows.

The MNN tool was written in the Java language using the JBuilder-4 development tool. The Java Runtime Environment is required for the MNN tool to function. The MNN system consists of two main parts: the user interface and the neural network system. The user interface includes a number of screens for receiving input and displaying output. The neural network system includes several classes implementing methods for training and testing neural networks, as well as for making forecasts.

The main classes implemented in the neural network system are illustrated in Fig. 2. The synapse class calculates the weighted input of a neuron and performs weight changes. The neuron class includes functions to connect with another neuron in the network and to activate transfer functions. Several neurons are connected together to form a back-propagation neural network (BPNN) which is also a multi-layer perceptron. The BPNN class implements methods for training, testing and forecasting. If the multiple-step-ahead validation mode is triggered, an individual network communicates with other lower-ordered neural networks during the training process. There is also a class that manages all the neural networks in an array. This class communicates with the data-set class and activates the necessary methods in the component networks to conduct overall training, testing and forecasts. The data-set class reads time series' points from text files and creates

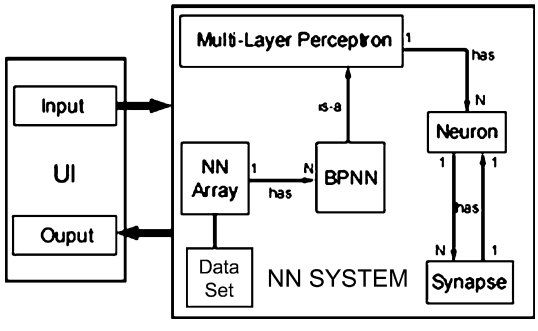

Fig. 2 Main classes of the neural network system of the MNN tool

training and testing records. The size of these records depends on user-input parameters and the topology of each component neural network.

All neural networks in the current MNN system are multi-layer neural perceptrons with only one hidden layer. The user can determine the neural network structures by setting the parameters including the number of input, output, hidden units, etc. The connection weights and biases are initialised with small random values.

The component neural networks in the system are trained with the back-propagation algorithm. The system repeats the training cycles until any one of the following scenarios occurs: the error reaches an acceptable threshold, the number of cycles reaches a pre-set maximum value, or over-fitting occurs.

Over-fitting happens when a neural network learns the training patterns well but has a poor generalisation ability. In our implementation, the MNN system determines that over-fitting has occurred when the validation error monotonically increases for a certain number of cycles.

Some sample screens from the MNN system are presented as follows.

The screen for inputting training parameters is shown in Fig. 3. Since most of the input items on the screen are self-explanatory, only brief additional explanations are provided as follows.

– Minimum number of training cycles: in most case this value is set to zero. However, in some cases the minimum number of training cycles needs to be set to a number greater than zero for the following reason. The validation error often increases at the beginning of the training process and then decreases eventually. However, the MNN tool may mistake the increasing error as over-fitting and halt the training. Enabling the user to set the minimum number of training cycles ensures the MNN tool would continue training past this period without stopping due to misperceived over-fitting.
– Maximum number of training cycles: when the neural network has not over-fitted the data and the validation error is still higher than the threshold set by users, the MNN system would continue training until the maximum number of cycles is reached.
– Validation interval: the validation interval is the interval between two consecutive validations in terms of training cycles. For example, if the validation interval is 4, validation is performed every 4 training cycles. The default value for this parameter is 1. Setting this parameter with a higher value reduces the necessary training time because validation errors are calculated less frequently during the training process.
– Validation window size: the validation window size is the number of consecutive non-decreasing validation errors needed before the system decides that over-fitting has occurred.
– Using multi-step validation: users can choose one-step validation or multi-step validation with this radio button. In one-step validation, the lead times for validation and training are the same. Each neural network is validated by itself. In multi-step validation, the lead times for validation spread over longer ranges (refer to Sect. 3). In the latter case, the training of higher-ordered neural networks requires the existence of trained lower-ordered neural networks to calculate the predicted output for a validation input vector. Hence, the training quality of the higher-ordered networks depends on the training quality of the lower-ordered neural networks.

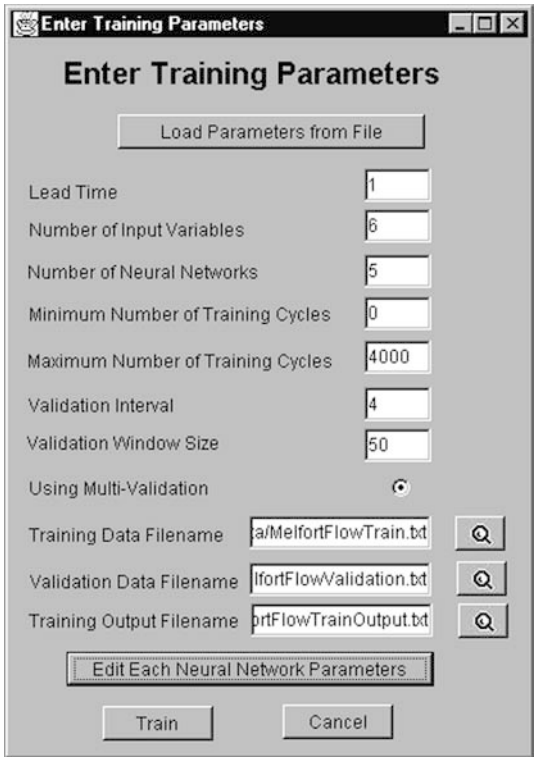

Fig. 3 A screen for inputting training parameters

When the user clicks the "Edit each neural network parameter" button in the screen shown in Fig. 3, the input screen for setting the training parameters of each individual neural network is displayed. Figure 4 shows the input screen for an 0-ordered neural network. After setting the parameters for one neural network, the user can click the "Next" button to advance to the next neural network. The major parameters on this screen as described as follows.

– Load neural network from file: the user can continue to train a previously trained neural network by selecting this radio button and entering the name of the neural network file.
– Neural network file name: when the "load neural network" radio button is checked, the filename for the neural network is entered here.
– Train neural network: since there are several neural networks in a MNN system, it is possible that the user

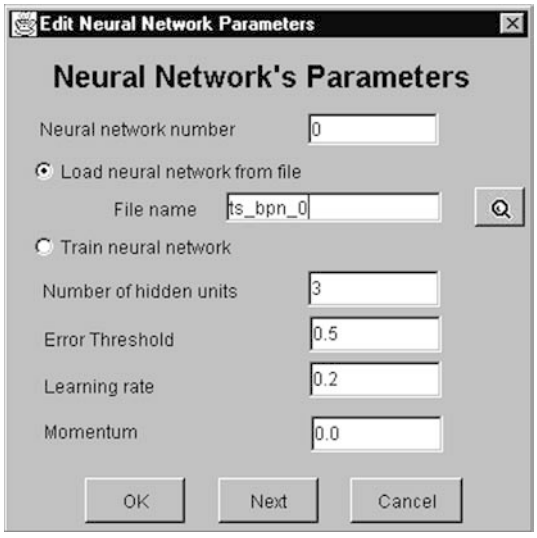

Fig. 4 A screen for inputting training parameters of the component neural network

wants to continue training only a particular component network. On the input screen for that particular network, the user can choose whether to train the component neural network with this radio button.
- Number of hidden units
- Error threshold (in percentage): when the validation error is less than or equal to this value, training stops.
- Learning rate: the learning rate is a scaling factor that decides how fast an algorithm should learn. A higher learning rate improves the learning speed. But if the rate is too high, the algorithm will exceed the optimum weights.
- Momentum: the momentum adds a contribution from the previous step when a weight is updated.

5 Case study

To illustrate the use of the single and multiple neural network models, the models are applied for the prediction of the future hourly gas flow through a compressor station in Saskatchewan. This station is a part of the gas pipeline distribution system at St. Louis East of Saskatchewan, Canada, and it is called the Melfort compressor station.

An overview schematic of the St. Louis East gas stations and their service areas is shown in Fig. 5. The system consists of two stations located at Melfort and St. Louis. The Melfort station receives gas from the St. Louis station and transmits it to the surrounding consumption areas of Nipawin and Hudson Bay. To satisfy customer demand at Nipawin and Hudson Bay, sufficient gas must be transmitted to the Melfort station and sufficient compressors must be running at the Melfort station. Therefore, dispatchers at the Melfort station need to make decisions to turn compressors on or off, or to adjust the compression level in order to maintain the necessary pressure while not wasting resources. The dispatcher's decision has a significant impact on effectiveness of the natural gas pipeline operation. When customer demand increases, a dispatcher adds compression to the pipeline system by turning on one or more compressors. On the other hand, the dispatcher turns off one or more compressors to reduce compression in the pipeline system when customer demand decreases. Incorrect decisions made

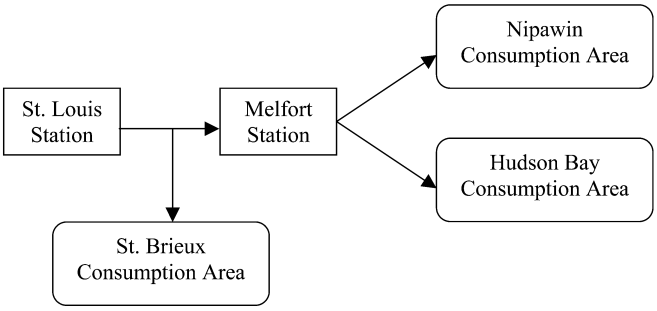

Fig. 5 The schematic of the St. Louis East system

by the dispatcher will cause substantial economic loss to the company.

The long term objective of this study is to aid the dispatcher in satisfying customer demand with minimal operating costs, i.e., to optimise natural gas pipeline operations. A short term objective is to predict customer demand for gas. A dispatcher needs to know ahead of time when the largest volume requirement of gas will occur and to be ready for it. Otherwise, the system pressures at Nipawin and Hudson Bay will be below the required minimum. Since gas consumption is estimated monthly from billing records, this estimated value cannot be used for predicting hourly demand. Instead, we used the flow rate at the Melfort station as a substitute variable for customer demand for gas.

An assumption made in the study is that flow rate at the Melfort station reflects the consumption patterns of customers at Nipawin and Hudson Bay. As illustrated in Fig. 6, the natural gas flow rate fluctuates during the day. For example, the demand is usually low at night. In the morning (from approximately 7 to 9 o'clock), the demand is higher because residential customers start showering and cooking and industrial customers start their machines. In the afternoon, the demand decreases since the facilities are already heated up. After work hours, the industrial customers' demand becomes lower while the residential customers' demand increases. The demand for natural gas also fluctuates depending on the season. In the winter, the demand for natural gas is usually higher than in the summer. Special occasions such as public holidays also constitute a factor that affects demand patterns.

5.1 Data collection and preprocessing

The data was obtained from SaskEnergy/Transgas. Hourly flow rates in the period from December 2001 to mid August 2002 were collected with an interruption from March 14th to May 27th, 2002. Since data from the fall (from September to November) and spring (from March to May) were not available, it was not possible to divide the data set into four seasonal data sets for separate treatments. The data was also preprocessed. Since there were several hourly flow rates with values of zero in the data set, these data points were assumed either missing or abnormal and were eliminated from the data set. Furthermore, since the sigmoid activation function was used which returns a number in the range of [0, 1], all hourly flow rates were normalised to this range. The following equation was adopted for normalisation:

$$x = (x - \min)/(\max - \min),$$

where min and max are the estimated minimum and maximum boundaries of monthly productions and not the actual boundaries in the training data set. By examining the plot of the historical data set, the min and max values were estimated as 0 and 600 ($10^3 m^3$). It can be observed that the sole purpose of the normalisation process is to ensure the range of output values from the neural network model corresponds to the range of outputs from the sigmoid function. Since the neural network method does not require stationary of data, a log correlation can be learnt and reflected in the neural network's weights.

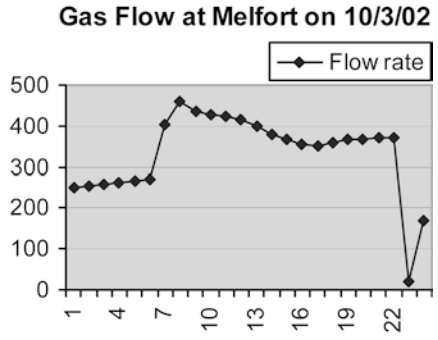

Fig. 6 Hourly flow during a day

Furthermore, the data set was divided into three subsets for training, validation and testing in the proportion of 5:1:1. The training set contains approximately 2500 data points. The validation and test sets contain around 500 data point each.

5.2 Training and validation

Some consideration for training and validating the neural network model are discussed as follows. It was determined that the size of the input vector would consist of six hours of data, which constitutes data for a quarter of a day. The rationale was that a shorter period may not contain enough information to predict 24 hours ahead while a longer period may render the neural networks too complex and therefore require more data to train. Since the amount of data available was limited, it is not desirable to have more inputs because it would result in more weights to tune and higher possibility to overfit the data.

The number of ANNs in the MNN was determined based on the length of the expected prediction term. Since $2^4 < 24 < 2^5$, a maximum of four ANNs were used to predict 24 hours ahead. However, this number can be set smaller based on validation errors produced by different combinations of ANNs.

The weights of the first ANN were initialised with small values in the range from 0 to 0.5. The subsequent ANNs were initialised with the previous ANN's weights in order to reduce training time.

Validation was done every four training cycles. Four was selected to reduce the validation time spent. It was observed that single step validation gave better results than multiple step validation in this application.

After five neural networks had been trained, their five combinations, which included 1, 2, 3, 4 and 5 neural networks, were validated on the validation set to predict 24 hours ahead. The one with the lowest validation error rate out of the last four was chosen as the final MNN. The one with only one neural network was the single ANN.

A comparison of results generated from running both the single and multiple neural network models is shown in Fig. 7. As can be seen in Fig. 7, the MNNs with 4 and 5 neural networks consistently performed better than the single ANN model. Meanwhile, the MNNs with 2 and 3 neural networks gave good results at first and became less and less effective when the prediction period became longer. The model with 5 neural networks performs only marginally better than the one with 4 neural networks. The average MAPEs for the five models with 1 to 5 neural networks were 11.7%, 40.3%, 11.02%, 8.84% and 8.76%, respectively. The one with 5 neural networks was chosen as the final MNN for testing.

5.3 Testing

In order to facilitate a comparison between a MNN and a single ANN, the same set of test data was applied to the MNN and the single ANN to predict hourly flow rate for different leads from 1 to 24 hours. The results are summarised in Fig. 8.

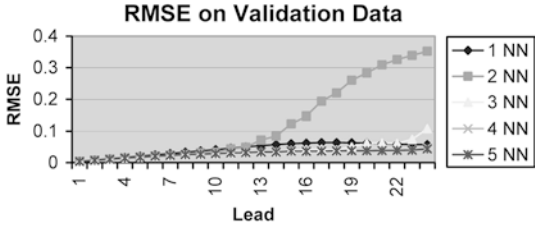

Fig. 7 The validated RMSE of 5 models for a 24 hour period

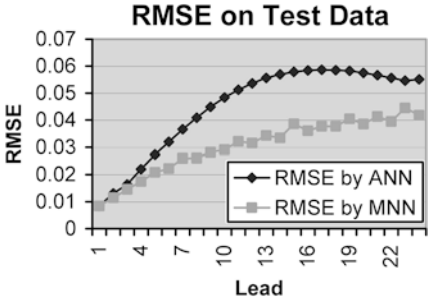

Fig. 8 Test errors for the MNN and the single ANN for a 24 hour period

It can be seen from Fig. 8 that the MNN consistently performs better than the single first ordered ANN. As the prediction term increases, the difference is more significant. This indicates a MNN performs better than a single ANN in long term forecast.

The $MAPE(i)$ is the mean absolute percentage error for lead i, and the average MAPEs were calculated as follows:

$$\text{Average}MAPE = \frac{1}{24}\sum_{i=1}^{24} MAPE(i)$$

For a 24 hour prediction, the average errors were 12.38% with the single ANN and 8.736% with the MNN. The desired and predicted outputs from the MNN and ANN model for a prediction lead of 24 hours are shown in Fig. 9.

As can be seen, the MNN model performs better than the ANN model. The predictions generated by the MNN shaped like a delayed version of the actual outputs, but the range of values approximate the actual values. On the other hand, the prediction generated by the single ANN were rather random. However, neither model performed completely satisfactorily on the 24-hour prediction; this can be explained as follows. Firstly, there may not be enough data points for training. The data used in this study was collected in less than a year. Secondly, special occasions such as holidays and weekends were not considered as factors for prediction. The gas usage patterns on such occasions may be different from that of a normal day. Thirdly, seasonal effects may play a role but they were not considered either. A network trained on a data set for summer may not generalise well in winter. Future research could include collecting more data and a meaningful decomposition of the problem into sub-problems. Data classification could be based on the seasons, and weekends versus weekdays.

Predictions of 6 hours ahead were also made because the amount of available data was limited. For the 6 hours ahead prediction, the average errors were 5.75% with the single ANN and 4.971% with the MNN. An error of 5% or less was considered acceptable. The predicted outputs from MNN and ANN for a prediction lead of 6 hours are shown in Fig. 10.

As can be seen in Fig. 10, both predicted lines are quite close to the actual lines but the MNN model achieved better results than the single ANN model.

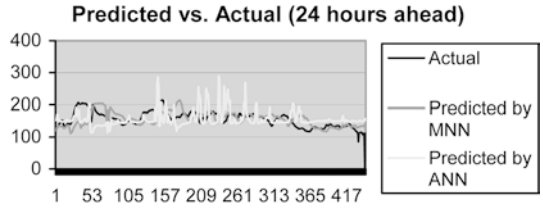

Fig. 9 Predicted vs. actual for 24 hours ahead

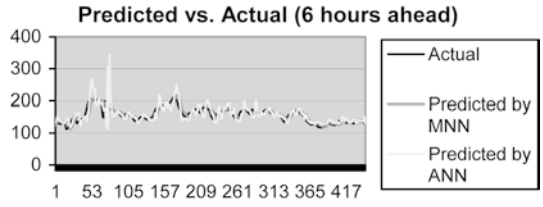

Fig. 10 Predicted vs. actual for 6 hours ahead

6 Conclusions and future works

The application case study shows that a MNN model shows superior performance over a single ANN model in long-term prediction. However, if the period is too long, neither model can predict well. Incorporating more neural networks into the model does not guarantee that the error would be lowered. As it can be seen in the application case study, the model with two neural networks did not perform more satisfactorily than the single neural network. Similarly, using more than five neural networks to predict a 24 hour period ahead is not useful because the neural networks with orders greater than or equal to five predict 32 periods or above. The accuracy of the model will not improve even if a higher ordered neural network was used.

A weakness of the MNN technique is that a data set involving longer and more continuous time series is needed in order to build the model.

Furthermore, some limitations that cause inaccuracy in forecasting no matter how well a model is trained can be observed as follows:

- There is genuine random noise in the data due to errors in recording the data. In the time series, we noticed several non-zero but abnormal data points. It is not known whether they were incorrectly recorded or if they are irregularities in the data set.
- The sample data set does not spread evenly in the feature space and constitutes only a partial representation of the population. In the time series context, this means that the length of time series under investigation is insufficient to represent all patterns in the problem space. For example, the data set in the gas consumption time series does not include data on some seasons, which means the seasonal factor cannot be considered. It is difficult to make any improvement unless more data is collected. With the currently available amount of data, an ARIMA model can be an alternative method.
- The factors that significantly influence the variable to be forecasted are either completely unavailable or partially available within the examined time span. An example of such parameters is temperature. When the temperature declines, customers tend to use more gas. However, the parameter of temperature is not in the data set and future temperatures themselves is hard to predict. Including temperature as one input parameter for the model could be an item in the future research agenda.

A weakness in the reported work is that only the simple hold out validation method is employed in the current systems. The data set was divided into two portions for training and testing. However, this method of evaluation can have a high variance. The evaluation may depend heavily on which data points are included in the training set and which are included in the test set. Thus, the evaluation results may be significantly different if a different division of data is adopted. Cross-validation could be used in the future.

The current MNN tool still needs much improvement. Considering the serious loss in the number of data records when a missing data point is eliminated, a method to replace the missing data should be used. A simple approach to be used in the future is to replace the missing data point with the average of neighbouring points.

A future topic for research could be developing a more automatic strategy for training which can reduce users' efforts. A number of methods for adapting the learning rate such as a bold driver [13] and annealing [1] have been proposed in the literature. In a bold driver neural network, after each training cycle, the training error is compared to its previous value. If the error has decreased, the learning rate is increased slightly. If the error has increased significantly, the last weight changes are discarded and the learning rate is decreased sharply. The bold driver method keeps increasing the learning rate slowly until it finds itself taking a step that has clearly gone too far up the opposite slope of the error function. The annealing method gradually lowers the global learning rate.

Furthermore, a more ambitious improvement to the MNN system would be to design and implement different kinds of training algorithms for the component neural networks. Other possible topics for future research are expanding the set of input variables to include temperature data, and comparing the MNN model with a multi-process dynamic linear model.

Acknowledgements We are grateful to SaskEnergy/Transgas for the data set and their support for this project. The generous support of a strategic grant from NSERC is also gratefully acknowledged. The authors would also like to thank the anonymous referees for their constructive comments.

References

1. Bos S, Amari S (1998) Annealed online learning in multilayer neural networks. J Phys A Math Gen 31(22):L413-L417
2. Cho SB, Kim JH (1995) Combining multiple neural networks by fuzzy integral for robust classification. IEEE Trans Sys Man Cybern 25(2):380–384
3. Darbellay GA, Slama M (2000) Forecasting the short-term demand for electricity: do neural networks stand a better chance? Int J Forecast 16:71–83
4. Dorffner G (1996) Neural networks for time series processing. Neur Ntwk Wor 6(4):447–468
5. Duhoux M, Suykens JAK, De Moor B and Vandewalle J (2001) Improved long-term temperature prediction by chaining of neural networks. Int J Neur Sys 11(1):1–10
6. Faraway J, Chatfield C (1998) Time series forecasting with neural networks: a comparative study using the airline data. Appl Stat 47:231–250
7. Hashem S, Schemeiser B and Yih Y (1994) Optimal linear combinations of neural networks: an overview. In: Proceedings of the 1994 IEEE International Conference on Neural Networks (ICNN'94), vol. 3, Orlando, FL, June 1994
8. Kadaba N, Nygard KE, Juell PL and Kangas L (1989) Modular back-propagation neural networks for large domain pattern classification. In: Proceedings of the International Joint Conference on Neural Networks (IJCNN'89), Washington DC, 18–22 June 1989
9. Lertpalangsunti N, Chan CW (1998) An architectural framework for construction of hybrid intelligent forecasting systems: application for electricity demand prediction. Engin App Art Intellig 11:549–565
10. Lertpalangsunti N, Chan CW, Mason R and Tontiwachwuthikul P (1999) A toolset for construction of hybrid intelligent forecasting systems: application for water demand prediction. Art Intellig Engin 13(1):21–42
11. Lee BJ (1996) Applying parallel learning models of artificial neural networks to letters recognition from phonemes. In: Proceedings of the Conference on Integrating Multiple Learned Models for Improving and Scaling Machine Learning Algorithms, Portland, OR, 4–5 August 1996
12. McNames J, Suykens JAK and Vandewalle (1999) Winning Entry of the K. U. Leuven Time Series Prediction Competition, Int J Bifurc Chaos 9(8):1485–1500
13. Sarkar D (1995) Methods to speed up error back-propagation learning algorithm. ACM Comput Surv 27(4):519–542
14. Tang Z, Almeida C and Fishwick PA (1991) Time series forecasting using neural networks vs. Box-Jenkins methodology. Simulation 57(5):303–310
15. Walczak S (2001) An empirical analysis of data requirements for financial forecasting with neural networks. J Manage Info Sys 17(4):203–222